Department of Computer Science
National Tsing Hua University

CS 2336: Discrete Mathematics

Chapter 13

Optimization and Matching

(Overview)

Instructor: Cheng-Hsin Hsu

Operations Research

- Finding optimum solution under various constraints
- Examples related to graphs/multigraphs:
  - The shortest distance between two vertices in a loop-free undirected graph
  - The spanning tree with the minimum total weight
  - The maximum amount of material that can be transported from a source to a destination over a transport network
    - Material can be water, oil, and network packets
- We will cover a few popular algorithms

Outline

13.1 Dijkstra's Shortest-Path Algorithm

13.2 Minimal Spanning Trees: The Algorithms of Kruskal and Prim

13.3 Transport Networks: The Max-Flow Min-Cut Theorem

13.4 Matching Theory

Weighted Graph

- For a loop-free connected directed graph G=(V,E), we assign a weight wt(e) to each of the edge e=(a,b), where a and b are two vertices. G is called a weighted graph.
  - wt(e) is a real number, and can also be written as wt(a,b)
  - wt(x,y) is infinity if (x,y) is not an edge in G

- Ex 13.1: The weights can represent the driving distance, flying time, transportation cost from location x to y

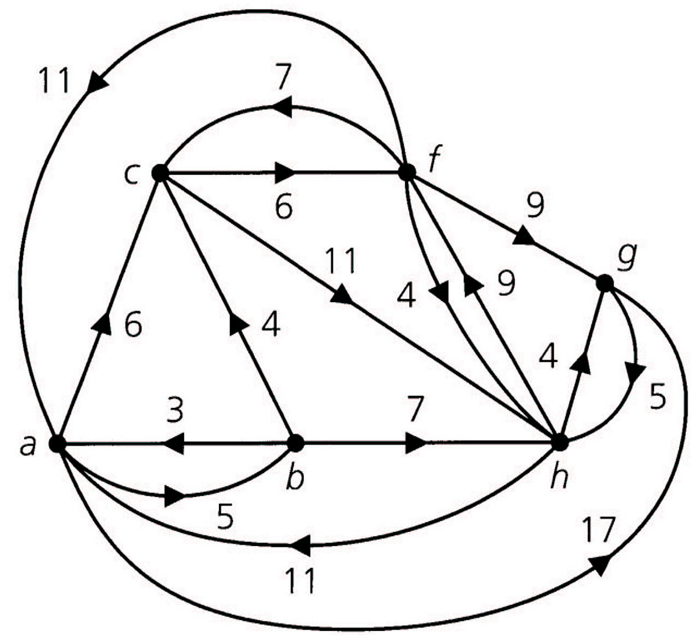


Figure 13.1

Shortest Path

- For a path $(a,v_1),(v_1,v_2),\ldots,(v_n,b)$, its length is defined as $wt(a,v_1)+wt(v_1,v_2)+\ldots+wt(v_n,b)$
- We define d(a,b) as the shortest distance from a to b, which is the length of the shortest path between them
- $d(a,b) = \infty$ if no such path exists, and d(a,a)=0
- Shortest path problem: Given a vertex v_0, for all vertex v in a graph, determine: (i) $d(v_0,v)$ and (ii) a directed path from v_0 to v if $d(v_0, v) \neq \infty$

Properties of *d* Function

- Let $S \subset V$, $v_0 \in S$, and $\bar{S} = V - S$. We define the distance from v_0 to $\bar{S}$ by: $d(v_0, \bar{S}) = \min_{v \in \bar{S}} d(v_0, v)$
- If $d(v_0, \bar{S})$ is finite, then there exist a directed path from v_0 to a vertex $v_{m+1} \in \bar{S}$, i.e., $d(v_0, \bar{S}) = d(v_0, v_{m+1})$
- Write the path as $P : (v_0, v_1), (v_1, v_2), \ldots, (v_{m-1}, v_m), (v_m, v_{m+1})$
- We have:
  - $v_0, v_1, \ldots, v_m \in S$
  - $P' : (v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ is the shortest path from v_0 to v_k, for all $1 \leq k \leq m$

Properties of *d* Function (cont.)

- In summary, given: $d(v_0, \bar{S}) = \min\{d(v_0, u) + wt(u, w)\}$
- Let $u^* \in S$ and $w^* \in \bar{S}$ lead to $d(v_0, \bar{S})$. We know the shortest distance from v_0 to w^* is $d(v_0, w^*) = d(v_0, u^*) + d(u^*, w^*)$
- This is the core idea of Dijkstra's shortest path algorithm

Dijkstra's Algorithm

- Let G=(V,E) be the weight graph with |V|=n. Find the shortest distance from v_0 to all other vertices
- Step 1: Let $S_0=\{v_0\}$, i=0. Label v_0 with (0,-) and all other vertices with $(\infty, -)$, where the 1st element is the shortest distance known so far, and the 2nd element is the previous vertex of the shortest path
- Step 2: For each $v \in \bar{S}_i$, update the label of v by (L(v), y), where y is the vertex in S_i producing the minimum L(v) and $L(v) = \min_{u \in S_i}\{L(v), L(u) + wt(u,v)\}$

Dijkstra's Algorithm (cont.)

- Step 3: If all vertices in $\bar{S}_i$ have label $(\infty, -)$ then stop. Otherwise
  - find a vertex v_{i+1}, where $L(v_{i+1})$ is minimum among vertices in $\bar{S}_i$
  - Let $S_{i+1} = S_i \cup \{v_{i+1}\}$
  - Let $i = i + 1$, stop if i=n-1, otherwise, go to step 2

- Once the algorithm is completed, the shortest path to any vertex v can be found by going reservedly toward v_0, following the labels

Simple Example, Iteration 0

- Ex 13.2: Find the shortest path from c to all other vertices using Dijkstra's algorithm

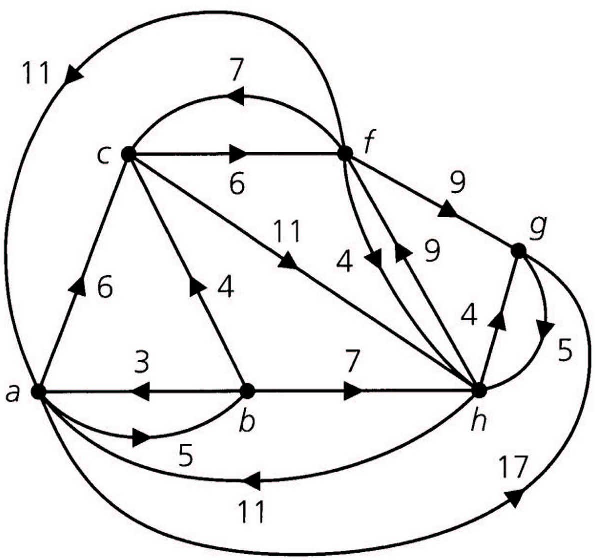
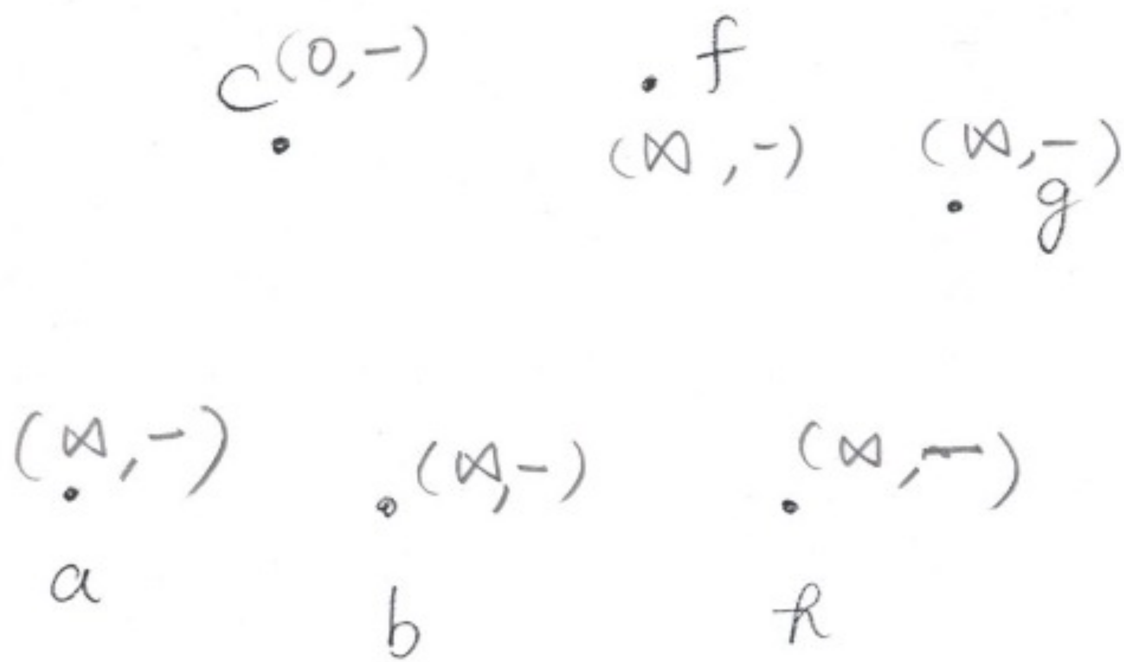


Figure 13.1



Simple Example, Iteration 1

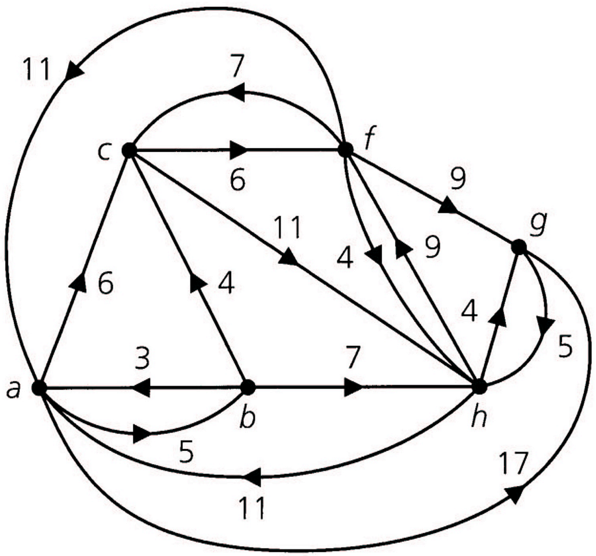
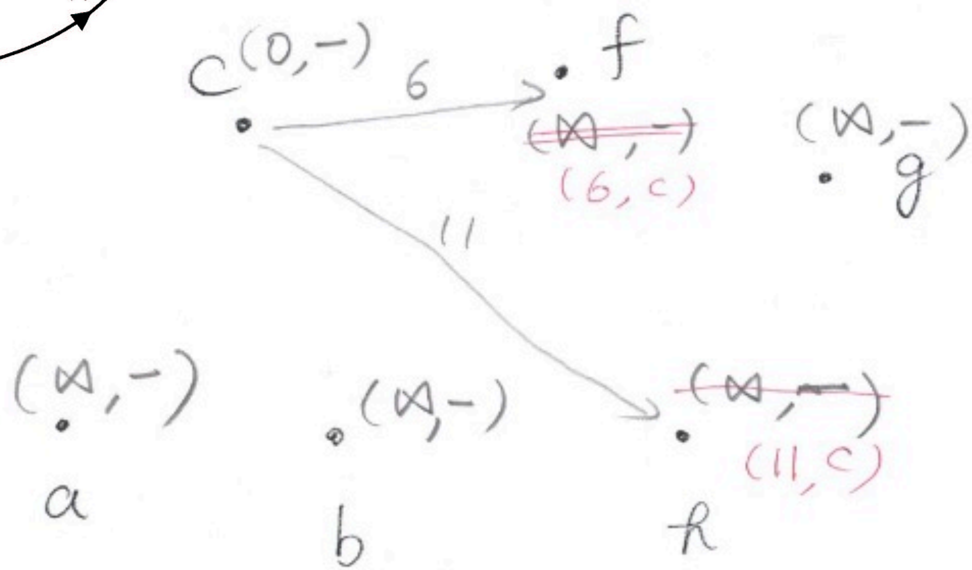


Figure 13.1

$S_2 = \{ c, f \}$



Simple Example, Iteration 2

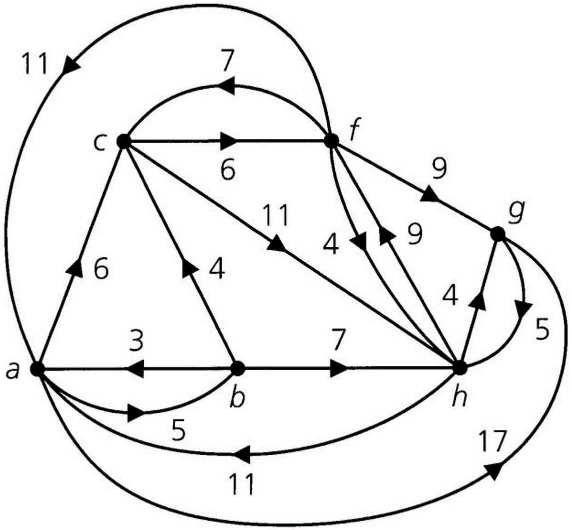
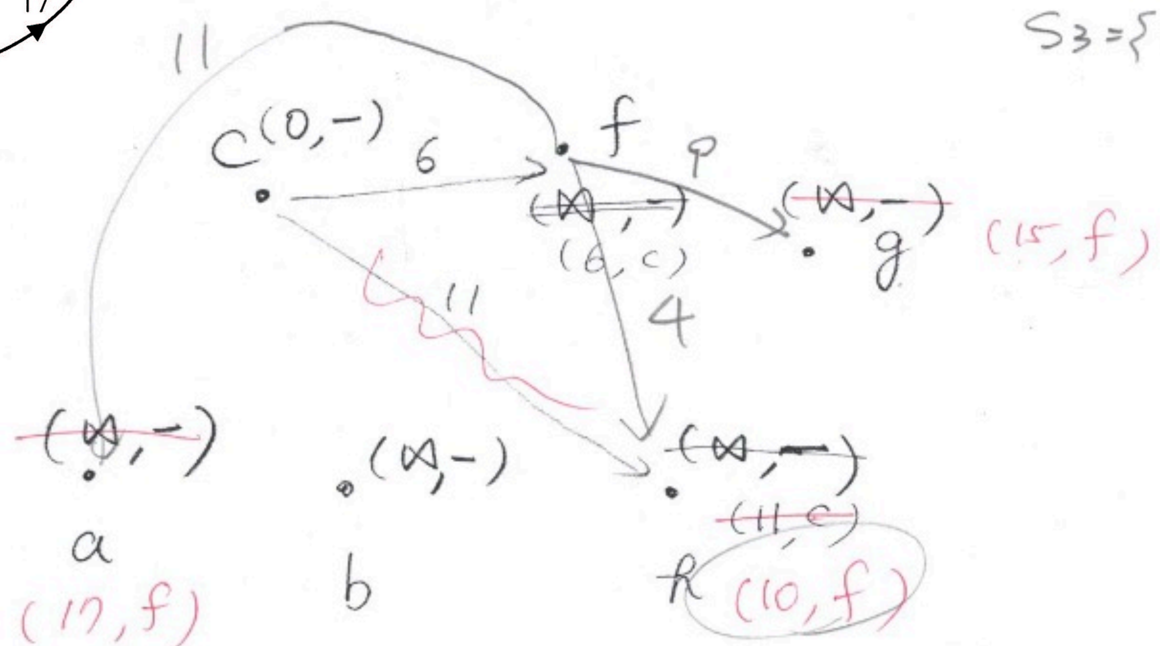


Figure 13.1

~~$S_2 = \{c, f\}$~~

$S_3 = \{c, f, h\}$



Simple Example, Iterations 3&4

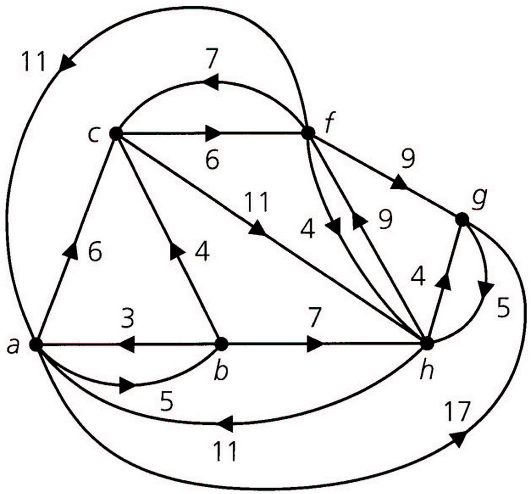
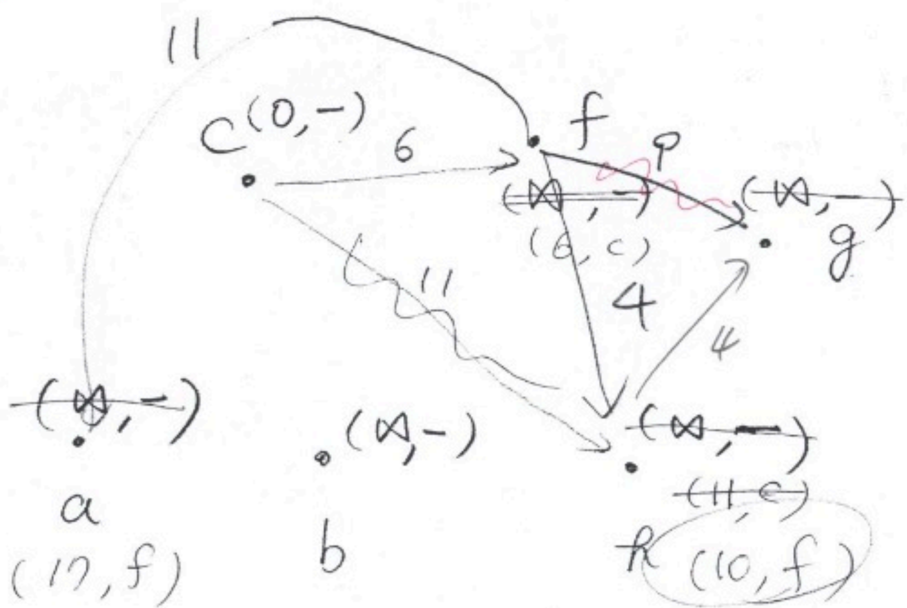


Figure 13.1



~~$S_2 = \{c, f\}$~~

~~$S_3 = \{c, f, h\}$~~

$S_4 = \{c, f, h, g\}$

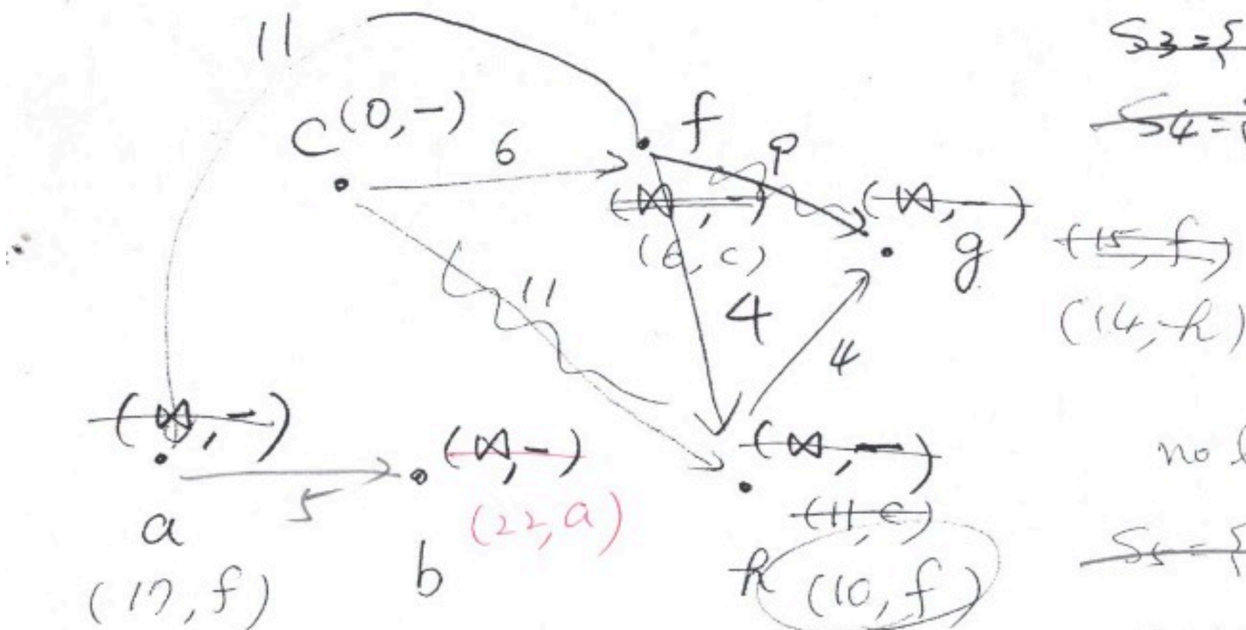
~~(15, f)~~

(14, h)

no label changed

$S_5 = \{c, f, h, g, a\}$

Simple Example, Iteration 5



~~S2 = {c, f}~~

~~S3 = {c, f, h}~~

~~S4 = {c, f, h, g}~~

no label changed

~~S5 = {c, f, h, g, a}~~

S6 = {c, f, h, g, a, b}

V	d(c, V)	path
a	17	c, f, a
b	22	c, f, a, b
c	0	c
f	6	c, f
g	14	c, f, h, g
h	10	c, f, h

Complexity Analysis

- Standard Dijkstra's algorithm has a time complexity of $O(n^3)$, where n is the number of vertices
  - Each time we add one vertex into S $\leftarrow$ first n
  - For every vertex, we check if we need to update the label, the other two n's
- Optimized Dijkstra's algorithms have complexities of $O(n^2)$ and $O(m \log n)$, where m is the number of edges
- Important: Dijkstra's is a greedy algorithm. Each step only depends on local information. However, the resulting solution achieves global optimum.

Outline

13.1 Dijkstra's Shortest-Path Algorithm

13.2 Minimal Spanning Trees: The Algorithms of Kruskal and Prim

13.3 Transport Networks: The Max-Flow Min-Cut Theorem

13.4 Matching Theory

Optimum Spanning Tree

- If we need to connect 7 computers, while connecting any two computers, x and y, imposes a cost wt(x,y). Find the spanning tree with the minimum total construction cost.
- We introduce two algorithms: Kruskal's and Prim's algorithms

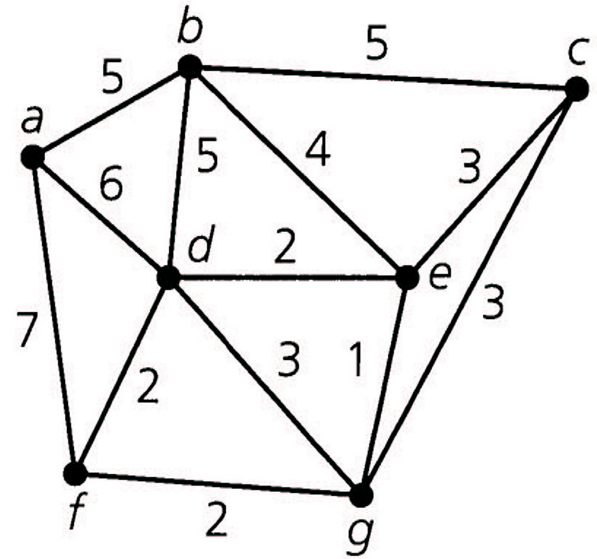


Figure 13.5

Kruskal's Algorithm

- Step 1: Let $i=1$, and select an edge e_1 in G with the smallest weight $wt(e_1)$
- Step 2: Let $e_1,e_2,\ldots,e_i$ be all the selected edges. Select edge e_{i+1} so that: (i) $wt(e_{i+1})$ has the smallest weight and (ii) the subgraph $e_1,e_2,\ldots,e_i,e_{i+1}$ contains no cycles
- Step3: Let $i=i+1$. If $i<n-1$, go to step 2. If $i=n-1$, subgraph given by $e_1,e_2,\ldots,e_{n-1}$ is an optimal spanning tree

Simple Example

- Ex 13.3: Find the optimum spanning tree of the graph using Kruskal's algorithm

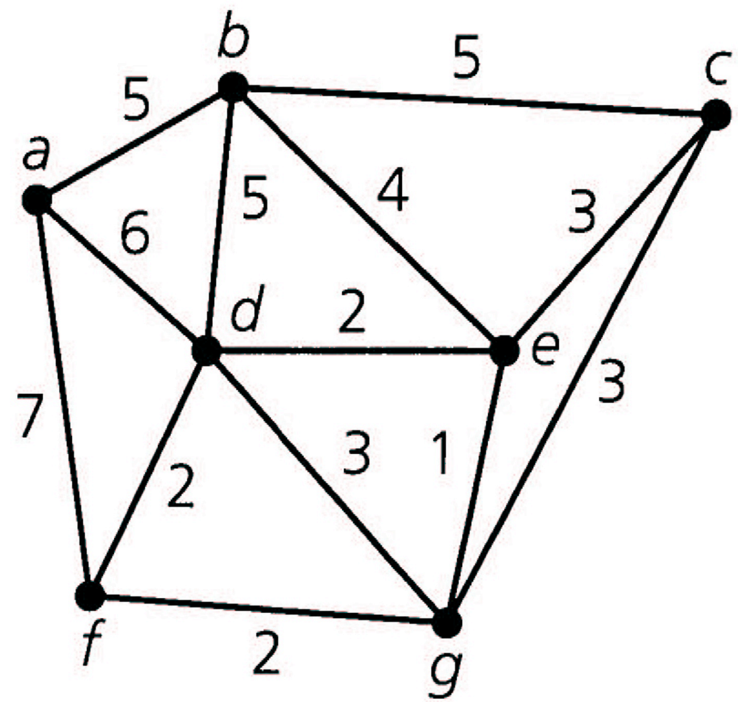


Figure 13.5

Simple Example (cont.)

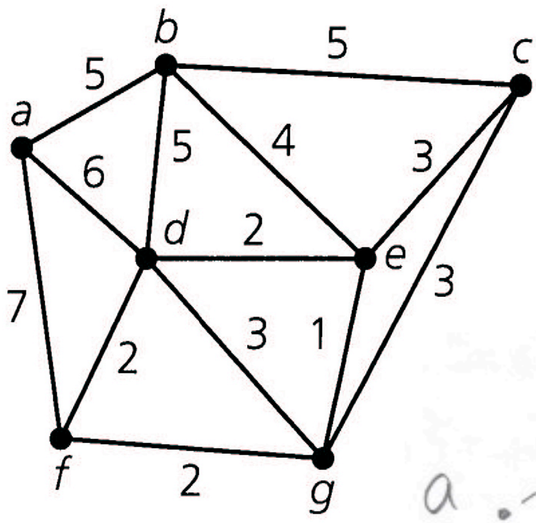
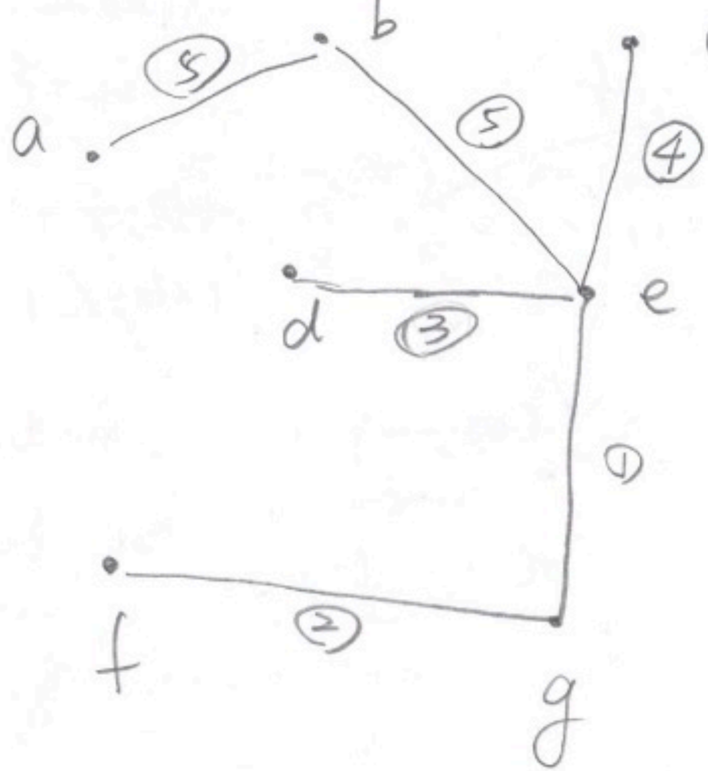


Figure 13.5



$5+4+3+2+1+2$

$=17$

Prim's Algorithm

- Step 1: Let i=1, P={v_1}, where v_1 is an arbitrary vertex. Define N=V-{v_1} and T is empty set
- Step 2: when 0<i<n, where |V|=n. Let P={$v_1,v_2,\ldots,v_i$}, T={$e_1,e_2,\ldots,e_{i-1}$},and N=V-P. Add to T an edge with minimal weight (e_i) that connects a vertex x in P with a vertex v_{i+1} in N. Move v_{i+1} from N to P.
- Step 3: Let i=i+1. If i<n goto step 2. If i=n, T gives the optimal spanning tree

Simple Example

- Ex 13.4: Find the optimum spanning tree of the graph using Prim's algorithm

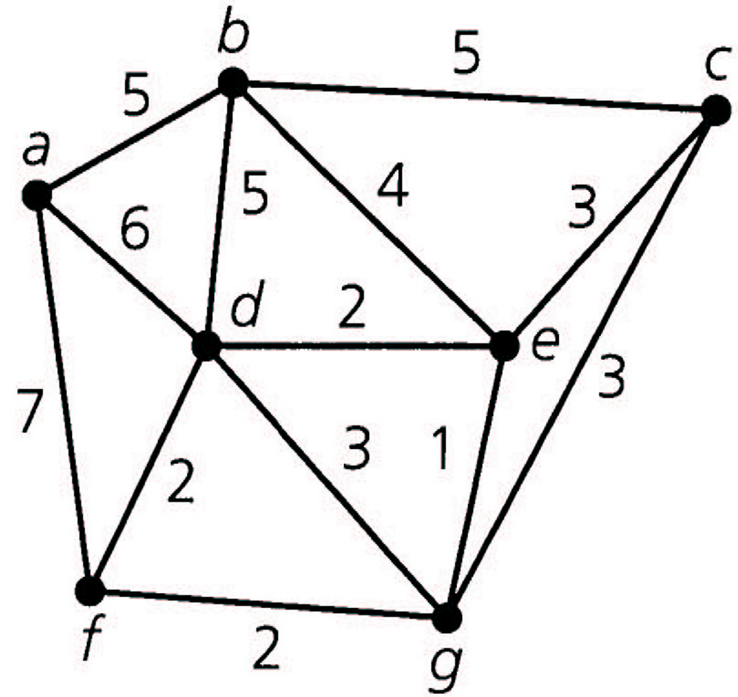


Figure 13.5

Simple Example (cont.)

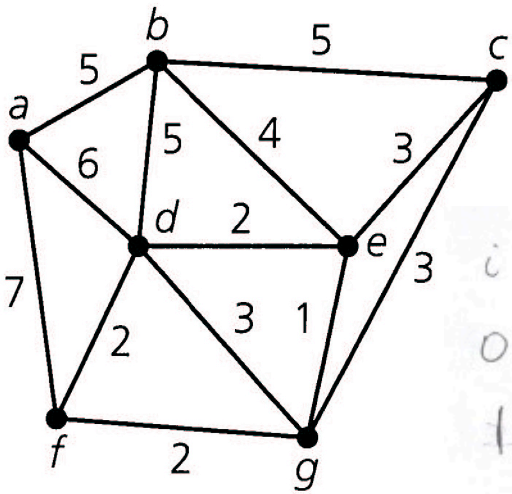


Figure 13.5

i	P	N	T
0	{a}	{b,c,d,e,f,g}	{ }
1	{a,b}	{c,d,e,f,g}	{{a,b}}
2	{a,b,e}	{c,d,f,g}	{{a,b}, {b,e}}
3.	{a,b,e,g}	{c,d,f}	{{a,b}, {b,e}, {e,g}}
4.	{a,b,e,g,f}	{c,d}	{{a,b}, {b,e}, {e,g}, {g,f}}
5	{a,b,e,g,f,d}	{c}	{{a,b}, {b,e}, {e,g}, {g,f}, {f,d}}
6.	{a,b,e,g,f,d,c}	{ }	{{a,b}, {b,e}, {e,g}, {g,f}, {f,d}, {c,e}}

$5+4+1+2+3+2=17$

Concluding Remark

- Both Kruskal's and Prim's algorithms always grow to optimal spanning trees
- Kruskal's algorithm may lead to forests during an iteration, while Prim always gives trees
- Prim allows us to start from any vertex
- Both algorithms are greedy, yet achieve global optimum

Outline

13.1 Dijkstra's Shortest-Path Algorithm

13.2 Minimal Spanning Trees: The Algorithms of Kruskal and Prim

13.3 Transport Networks: The Max-Flow Min-Cut Theorem

13.4 Matching Theory

Transport Networks

- N=(V,E) is a loop-free connected directed graph. N is called a (transport) network, if the following conditions are met
  - There is a unique vertex a, with id(a)=0. a is called the source
  - There is a unique vertex z, with od(a)=0. z is called the sink
  - N is weighted. There is a capacity function maps each edge e=(v,w) to a nonnegative integer, denoted by c(e)=c(v,w)

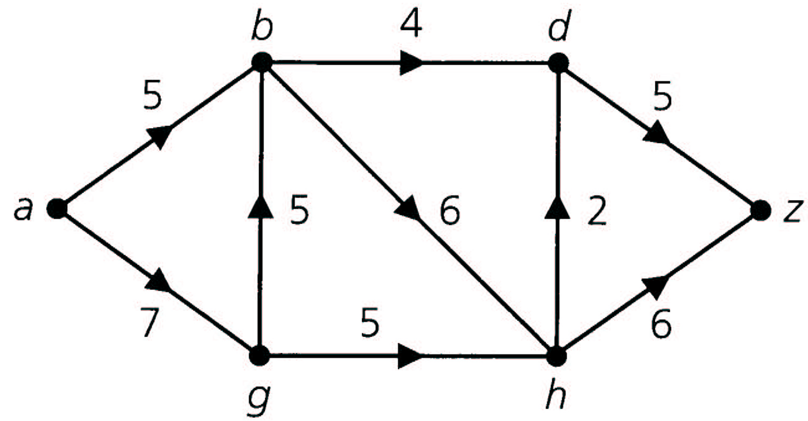


Figure 13.9

Flow

- N=(V,E) is a transport network. A function f from E to the nonnegative integers is called a flow for N if
  - F(e)<=c(e) for each edge e
  - For v other than the source and sink, $\sum_{w \in V} f(w,v) = \sum_{u \in V} f(v,u)$

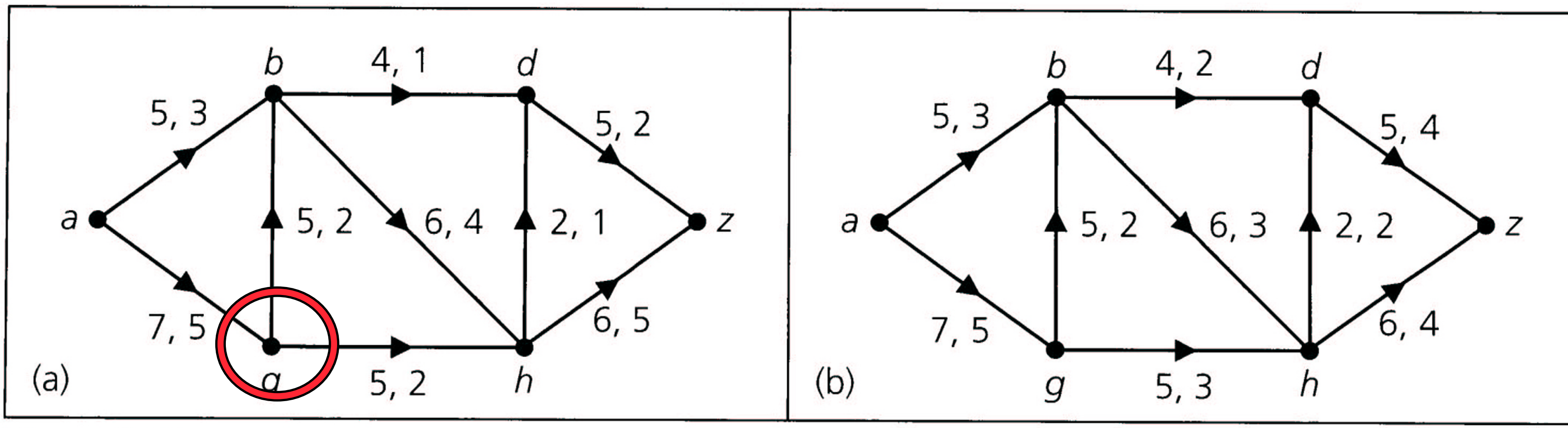
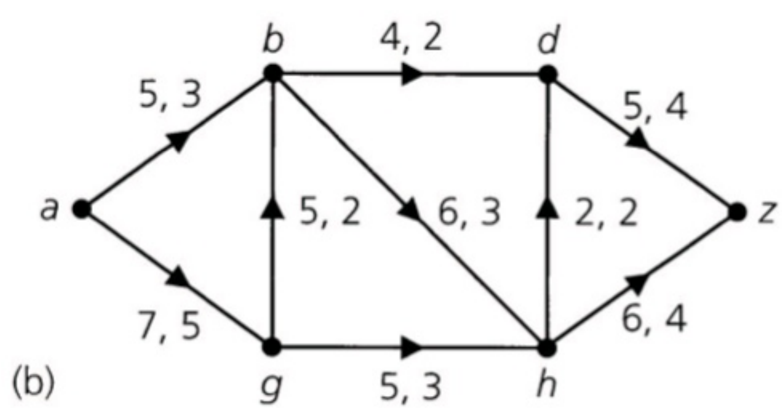


Figure 13.10

Value of a Flow

- Let f be a flow of network N=(V,E)
  - An edge e is saturated if f(e)=c(e), o.w. it is called unsaturated
  - If a is the source of N, $\mathrm{val}(f) = \sum_{v \in V} f(a,v)$ is called the value of the flow
- If z is the sink, not hard to see that $\mathrm{val}(f) = \sum_{v \in V} f(v,z)$
- Between source and sink?



Cut

- For network N=(V,E), C is a cut-set for the undirected graph associated with N, then C is called a cut if removing the edges in C from N separates a and z.
- The sum of all the edges' capacity is called the capacity of a cut.

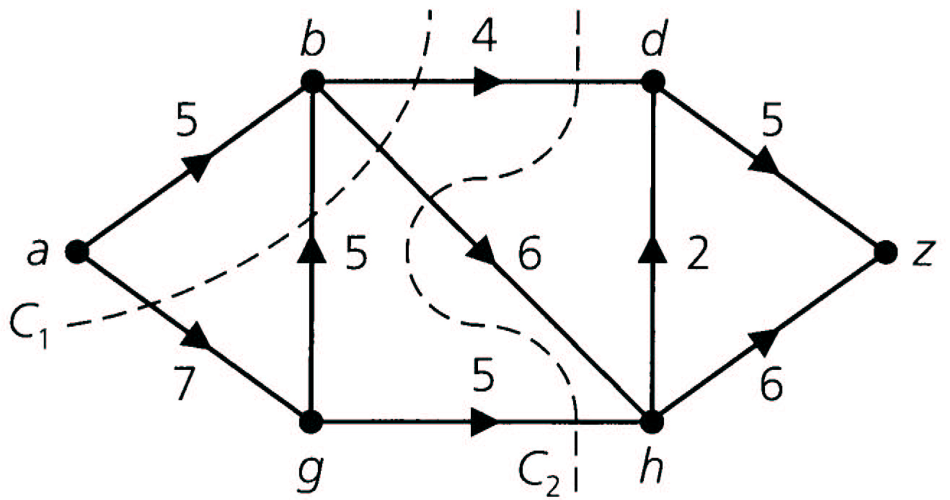


Figure 13.11

Cut Bounds Flow Value

- Val(f) cannot exceed the capacity of any cut in N.

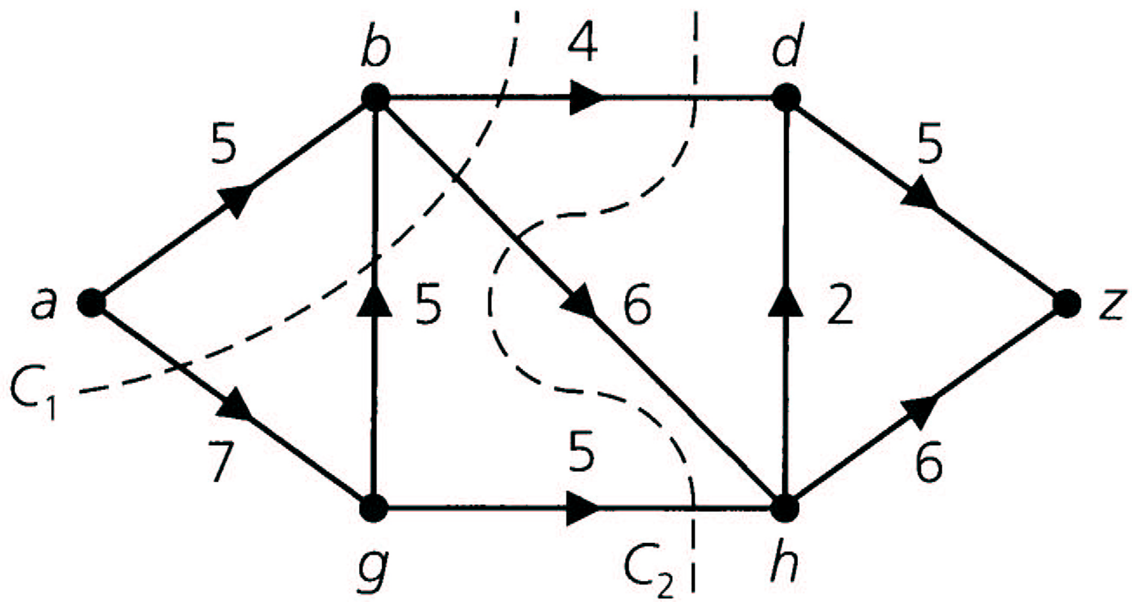


Figure 13.11

Max-Flow Min-Cut Theorem

- For a transport network N=(V,E), the maximum flow value that can be attained in N is equal to the minimum capacity over all cuts in the network

- Several algorithms have been proposed to solve max-flow min-cut problem. See the textbook for details.

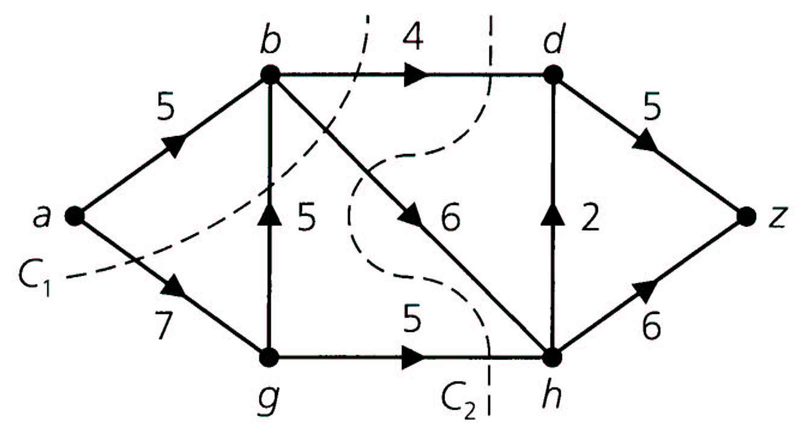


Figure 13.11